

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号
特開2002-82926
(P2002-82926A)

(43) 公開日 平成14年3月22日 (2002.3.22)

| (51) Int.Cl. ⁷ | 識別記号 | F I | テーマコード* (参考) |
|-------------------------------|-------|----------------|-------------------|
| G 0 6 F 15/177 | 6 7 8 | G 0 6 F 15/177 | 6 7 8 H 5 B 0 4 2 |
| 9/06 | Z E C | 9/06 | Z E C 5 B 0 4 5 |
| 11/36 | | 11/28 | 3 4 0 A 5 B 0 7 6 |
| 11/28 | 3 4 0 | 11/34 | S |
| 11/34 | | 9/06 | 6 2 0 R |
| 審査請求 未請求 請求項の数18 O L (全 15 頁) | | | |

(21) 出願番号 特願2000-270341 (P2000-270341)

(22) 出願日 平成12年9月6日 (2000.9.6)

(71) 出願人 000004226

日本電信電話株式会社
東京都千代田区大手町二丁目3番1号

(72) 発明者 田中 博樹

東京都千代田区大手町二丁目3番1号
日本電信電話株式会社内

(74) 代理人 100058479

弁理士 鈴江 武彦 (外2名)

Fターム(参考) 5B042 GA12 HH11

5B045 BB49 BB50 JJ08

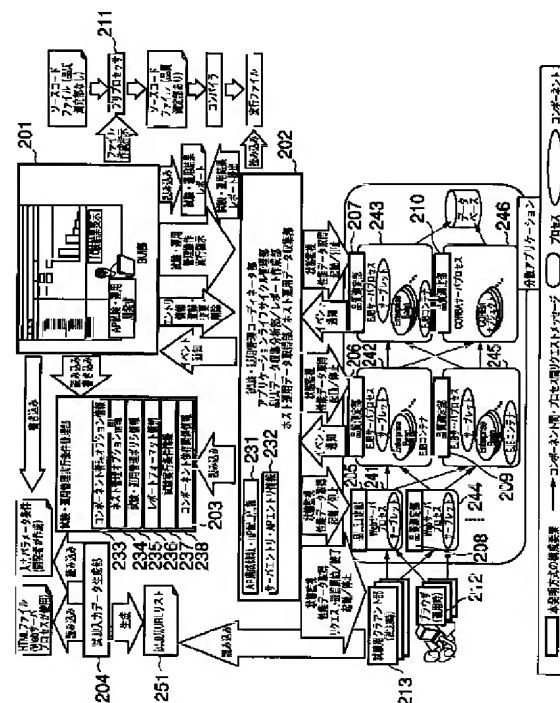
5B076 EC05

(54) 【発明の名称】 分散アプリケーション試験・運用管理システム

(57) 【要約】

【課題】 分散A Pの構成要素であるサーバプロセスやコンポーネント単位で信頼性、性能の評価、ボトルネックの特定をすることにより分散アプリケーション単位での試験・運用管理を行うことを可能とする。

【解決手段】 分散アプリケーションのソースコードファイル群にコンポーネントでの性能データを測定する品質測定部(313, 316, 319)を埋め込み、その後それらのソースコードファイルをコンパイラに通して、サーバプロセスを起動・動作させるために必要なサーバ実行ファイルを作成して動作させ、品質データ収集分析部(306)がその品質測定部から個々のコンポーネントの性能データを収集し(382-384)、またアプリケーションライフサイクル管理部(305)から正常動作性のデータを収集する(392)。



【特許請求の範囲】

【請求項1】 ソフトウェア部品であるコンポーネント間あるいはそのコンポーネントを内包するプロセス間の通信を実行するコンポーネント間通信プラットフォームあるいはプロセス間通信プラットフォームを一種以上含む分散アプリケーション動作環境を用い、該システムは：コンポーネントやそのコンポーネントを内包するプロセスなどであるサーバを起動及び停止するために必要な情報をサーバエントリとして保持し、一つのサービスを提供する一つ以上のサーバの集合である分散アプリケーション単位での起動や停止を行うために必要なサーバエントリの集合をアプリケーションエントリとして保持し、そのアプリケーションエントリを用いて分散アプリケーションの起動や停止を制御するとともに、アプリケーションのリアルタイムな構成情報を保持するアプリケーションライフサイクル管理部と、サーバに他のコンポーネントあるいはプロセスから処理要求が渡されるとき及びサーバから処理結果が返されるときに、その処理の開始時刻と終了時刻、処理名、及びその処理を実行したサーバ名あるいはそのサーバがコンポーネントである場合はそのクラス名を含む処理性能データを取得し、サーバが他のサーバに処理を要求し処理結果を受けるときに、その処理要求発行時刻と処理結果受信時刻、要求した処理の名前、及びその処理要求発行先のサーバ名あるいはそのサーバがコンポーネントである場合はそのクラス名を含む応答性能データを取得する性能データ取得部と、該性能データ取得部が取得した処理性能データと応答性能データから、個々のサーバでの処理実行時間、処理実行頻度、処理要求応答時間などを算出し、それら各々について予め規定された閾値を超過した場合にイベントメッセージを発生させる品質測定部と、任意に選択された一つの分散アプリケーションを構成する個々のサーバに関する処理実行時間、処理実行頻度、処理要求応答時間などのデータを、そのデータを保持する品質測定部群から収集して提供し、そのデータとアプリケーション構成情報やサーバ間の依存関係情報から分散アプリケーションの性能品質確保のボトルネックとなっている個所を特定し、また、任意に選択された一つの分散アプリケーションを構成する個々のサーバで発生した、異常終了や異常状態への遷移などの異常事象に関する、時刻、プロセスID、コンポーネント名、コンポーネントクラス名などの情報をアプリケーションライフサイクル管理部や品質測定部から収集して提供し、分散アプリケーションの信頼性品質確保のボトルネックとなっている個所を特定する品質データ収集分析部と、を具備することを特徴とする分散アプリケーション試験・運用管理システム。

【請求項2】 請求項1記載の分散アプリケーション試験・運用管理システムにおいて、

前記品質測定部がEJB (Enterprise Java Beans) コンポーネントの処理性能データを収集するにあたり、

品質測定部の一部である性能データ取得部が、コンポーネント間通信プラットフォームあるいはプロセス間通信プラットフォームに内在し、前記品質測定部の一部がEJBコンポーネントに処理要求が届く直前とEJBコンポーネントから処理結果が出された直後に任意の処理を実行するインタセプタとして動作し、

EJBコンポーネントに処理要求が届く直前の時刻（処理開始時刻）、EJBコンポーネントから処理結果が出された直後の時刻（処理終了時刻）、実行した処理の名前、及びその処理を実行したEJBコンポーネント名あるいはそのEJBコンポーネントのクラス名を含む処理性能データを性能データ取得部が取得し、品質測定部が保持することを特徴とする、分散アプリケーション試験・運用管理システム。

【請求項3】 請求項1記載の分散アプリケーション試験・運用管理システムにおいて、

前記品質測定部がEJBコンポーネントの処理性能データを収集するにあたり、

品質測定部の一部である性能データ取得部が、スケルトンに埋め込まれており、

EJBコンポーネントへ他のコンポーネントやプロセスから渡される処理要求がスケルトンにある性能データ取得部を通った時刻（処理開始時刻）、EJBコンポーネントから他コンポーネントへ返される処理結果がスケルトンにある性能データ取得部を通った時刻（処理終了時刻）、実行した処理の名前、及びその処理を実行したEJBコンポーネント名あるいはそのEJBコンポーネントのクラス名を含む処理性能データを性能データ取得部が取得し、品質測定部が保持することを特徴とする、分散アプリケーション試験・運用管理システム。

【請求項4】 前記分散AP動作環境を用いつつ、前記品質測定部がEJBコンポーネントの処理性能データを収集するにあたり、

品質測定部の一部である性能データ取得部が、EJBコンテナに埋め込まれており、

EJBコンポーネントへ他のコンポーネントやプロセスから渡される処理要求がEJBコンテナにある性能データ取得部を通った時刻（処理開始時刻）、EJBコンポーネントから他コンポーネントに返される処理結果がEJBコンテナにある性能データ取得部を通った時刻（処理終了時刻）、実行した処理の名前、及びその処理を実行したEJBコンポーネント名あるいはそのEJBコンポーネントのクラス名を含む処理性能データを性能データ取得部が取得し、品質測定部が保持することを特徴とする請求項2記載の分散アプリケーション試験・運用管

理システム。

【請求項5】 請求項1記載の分散アプリケーション試験・運用管理システムにおいて、前記品質測定部がEJBコンポーネントの処理性能データを収集するにあたり、品質測定部の一部である性能データ取得部が、EJBコンポーネントに埋め込まれており、EJBコンポーネントに他のコンポーネントやプロセスから処理要求が渡された直後の時刻（処理開始時刻）、EJBコンポーネントから処理結果が返す直前の時刻（処理終了時刻）、実行した処理名、及びその処理を実行したEJBコンポーネント名あるいはそのEJBコンポーネントのクラス名を含む処理性能データを性能データ取得部が取得し、品質測定部が保持することを特徴とする、分散アプリケーション試験・運用管理システム。

【請求項6】 請求項1記載の分散アプリケーション試験・運用管理システムにおいて、前記品質測定部がサーブレットの処理性能データを収集するにあたり、品質測定部の一部である性能データ取得部が、コンポーネント間通信プラットフォームあるいはプロセス間通信プラットフォームに内在し、サーブレットに処理要求が届く前とサーブレットから処理結果が出された後に任意の処理を実行するインタセプタとして動作し、サーブレットに処理要求が届く直前の時刻（処理開始時刻）、サーブレットから処理結果が出された直後の時刻（処理終了時刻）、実行した処理の名前、及びその処理を実行したサーブレット名あるいはそのサーブレットのクラス名を含む処理性能データを性能データ取得部が取得し、品質測定部が保持することを特徴とする、分散アプリケーション試験・運用管理システム。

【請求項7】 請求項1記載の分散アプリケーション試験・運用管理システムにおいて、前記品質測定部がサーブレットの処理性能データを収集するにあたり、品質測定部の一部である性能データ取得部が、サーブレットに埋め込まれており、サーブレットに他のコンポーネントやプロセスから処理要求が渡された直後の時刻（処理開始時刻）、サーブレットから処理結果が返す直前の時刻（処理終了時刻）、実行した処理名、及びその処理を実行したサーブレット名あるいはそのサーブレットのクラス名を含む処理性能データを性能データ取得部が取得し、品質測定部が保持することを特徴とする、分散アプリケーション試験・運用管理システム。

【請求項8】 請求項1記載の分散アプリケーション試験・運用管理システムにおいて、前記品質測定部がサーブレットの処理性能データを収集するにあたり、品質測定部の一部である性能データ取得部が、スケルト

ンに埋め込まれており、

サーブレットへ他のコンポーネントやプロセスから渡される処理要求がスケルトンにある性能データ取得部を通った時刻（処理開始時刻）、サーブレットから他コンポーネントやプロセスへ返される処理結果がスケルトンにある性能データ取得部を通った時刻（処理終了時刻）、実行した処理の名前、及びその処理を実行したサーブレット名あるいはそのサーブレットのクラス名を含む処理性能データを性能データ取得部が取得し、品質測定部が保持することを特徴とする、分散アプリケーション試験・運用管理システム。

【請求項9】 請求項1記載の分散アプリケーション試験・運用管理システムにおいて、前記システムは、前記品質測定部、該品質測定部に含まれる性能データ取得部、該性能データ取得部からのイベントメッセージを受信可能な試験・運用管理コーディネータ部を具備しており、サーバが実行した各々の処理について性能データ取得部がその処理開始時刻と処理終了時刻を取得したときに、品質測定部がその処理の処理実行時間を算出し、その処理実行時間が規定された時間を超過した事象が規定された時間範囲で規定された回数発生した場合に、品質測定部が試験・運用管理コーディネータ部にその旨のイベントメッセージを送出し、そのイベントメッセージを受信した試験・運用管理コーディネータ部が予め指定された内容で対処手続きを進めることを特徴とする、分散アプリケーション試験・運用管理システム。

【請求項10】 請求項1記載の分散アプリケーション試験・運用管理システムにおいて、前記品質測定部がEJBコンポーネントの応答性能データを収集するにあたり、品質測定部の一部である性能データ取得部が、コンポーネント間通信プラットフォームあるいはプロセス間通信プラットフォームに内在し、EJBコンポーネントから処理要求が発行された直後とEJBコンポーネントに処理結果が届く直前に任意の処理を実行するインタセプタとして動作し、EJBコンポーネントから処理要求が発行された直後の時刻（処理要求発行時刻）、EJBコンポーネントに処理結果が届く直前の時刻（処理結果受信時刻）、実行した処理の名前、及びその処理を実行したEJBコンポーネント名あるいはそのEJBコンポーネントのクラス名を含む応答性能データを性能データ取得部が取得し、品質測定部が保持することを特徴とする、分散アプリケーション試験・運用管理システム。

【請求項11】 請求項1記載の分散アプリケーション試験・運用管理システムにおいて、前記品質測定部がEJBコンポーネントの応答性能データを収集するにあたり、

品質測定部の一部である性能データ取得部が、スタブに埋め込まれており、

EJBコンポーネントが他のサーバに発行する処理要求がスタブにある性能データ取得部を通った時刻(処理要求発行時刻)、EJBコンポーネントが他のサーバから受ける処理結果がスタブにある性能データ取得部を通った時刻(処理結果受信時刻)、要求した処理の名前、及びその処理要求発行先のサーバ名あるいはそのサーバがコンポーネントである場合はそのクラス名を含む応答性能データを性能データ取得部が取得し、品質測定部が保持することを特徴とする、分散アプリケーション試験・運用管理システム。

【請求項12】 請求項1記載の分散アプリケーション試験・運用管理システムにおいて、

前記品質測定部がEJBコンポーネントの応答性能データを収集するにあたり、

品質測定部の一部である性能データ取得部が、EJBコンポーネントに埋め込まれており、

EJBコンポーネントが他のサーバに処理要求を発行する直前の時刻(処理要求発行時刻)、EJBコンポーネントが他のサーバから処理結果を受信した直後の時刻(処理結果受信時刻)、要求した処理の名前、及びその処理要求発行先のサーバ名あるいはそのサーバがコンポーネントである場合はそのクラス名を含む応答性能データを性能データ取得部が取得し、品質測定部が保持することを特徴とする、分散アプリケーション試験・運用管理システム。

【請求項13】 請求項1記載の分散アプリケーション試験・運用管理システムにおいて、

前記品質測定部がサーバレットの応答性能データを収集するにあたり、

品質測定部の一部である性能データ取得部が、コンポーネント間通信プラットフォームあるいはプロセス間通信プラットフォームに内在し、サーバレットから処理要求が発行された後とサーバレットに処理結果が届く前に任意の処理を実行するインタセプタとして動作し、サーバレットから処理要求が発行された直後の時刻(処理要求発行時刻)、サーバレットに処理結果が届く直前の時刻(処理結果受信時刻)、実行した処理の名前、及びその処理を実行したサーバレット名あるいはそのサーバレットのクラス名を含む応答性能データを性能データ取得部が取得し、品質測定部が保持することを特徴とする、分散アプリケーション試験・運用管理システム。

【請求項14】 請求項1記載の分散アプリケーション試験・運用管理システムにおいて、

前記品質測定部がサーバレットの応答性能データを収集するにあたり、

品質測定部の一部である性能データ取得部が、サーバレットに埋め込まれており、

サーバレットが他のサーバに処理要求を発行する直前の

時刻(処理要求発行時刻)、サーバレットが他のサーバから処理結果を受信した直後の時刻(処理結果受信時刻)、要求した処理の名前、及びその処理要求発行先のサーバ名あるいはサーバがコンポーネントである場合はそのクラス名を含む応答性能データを性能データ取得部が取得し、品質測定部が保持することを特徴とする、分散アプリケーション試験・運用管理システム。

【請求項15】 請求項1記載の分散アプリケーション試験・運用管理システムにおいて、

前記品質測定部がサーバレットの応答性能データを収集するにあたり、

品質測定部の一部である性能データ取得部が、スタブに埋め込まれており、

サーバレットが他のサーバに発行する処理要求がスタブにある性能データ取得部を通った時刻(処理要求発行時刻)、サーバレットが他のサーバから受ける処理結果がスタブにある性能データ取得部を通った時刻(処理結果受信時刻)、要求した処理の名前、及びその処理要求発行先のサーバ名あるいはそのサーバがコンポーネントである場合はそのクラス名を含む応答性能データを性能データ取得部が取得し、品質測定部が保持することを特徴とする、分散アプリケーション試験・運用管理システム。

【請求項16】 請求項1記載の分散アプリケーション試験・運用管理システムにおいて、

前記システムは、前記品質測定部、該品質測定部に含まれる性能データ取得部、該性能データ取得部からのイベントメッセージを受信可能な試験・運用管理コーディネータ部を具備しており、

サーバが他のサーバに発行した各々の処理要求について、性能データ取得部がその処理要求発行時刻と処理結果受信時刻を取得したときに、品質測定部がその処理要求の応答時間を算出し、

その応答時間が規定された時間を超過した事象が規定された時間範囲で規定された回数発生した場合に、品質管理部が試験・運用管理コーディネータ部にその旨のイベントメッセージを送出し、

そのイベントメッセージを受信した試験・運用管理コーディネータ部が予め指定された内容で対処手続きを進めることを特徴とする、分散アプリケーション試験・運用管理システム。

【請求項17】 請求項1乃至16のいずれか1項に記載の分散アプリケーション試験・運用管理システムにおいて、

Webサーバがブラウザに返すHTMLファイルの内容からWebサーバでのサーバレット起動を要求するHTTPリクエストに対応するURL記述部分をリストアップする試験入力データ生成部と、

その結果得られたURLリストと試験実行者が与えた引数データを用いて試験HTTPリクエストを発生させる

試験用クライアント部を用いることを特徴とする、分散アプリケーション試験・運用管理システム。

【請求項18】 請求項1乃至17のいずれか1項に記載の分散アプリケーション試験・運用管理システムにおいて、

任意の2つサーバの間で、特定のアプリケーションに依存する（依存元と依存先のサーバが双方とも動作状態にあり、特定のアプリケーションを起動している場合にのみ依存関係が有効）形、あるいは特定のAPに依存しない（依存元と依存先のサーバが双方とも動作状態にある場合に依存関係が有効）形で、依存関係を設定し、依存関係が設定された場合には、依存先のサーバの終了が検出されたときに、依存元のサーバも終了させる試験・運用管理コーディネータ部を具備することを特徴とする、分散アプリケーション試験・運用管理システム。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、電子商取引サービスや情報配信サービスなどの各種情報通信サービス（以下、サービスと記す）の分野において、サーバレット [Java Servlet API specification, SUN Microsystems Inc., <http://java.sun.com/products/servlet/index.html>]、CORBA (Common Object Request Broker Architecture) [Object Management Group, "The Common Request Broker: Architecture and Specification", Rev.2.3, <ftp://omg.org/pub/docs/formal/99-12-01.pdf>]、EJB (Enterprise Java Beans) [Enterprise JavaBeans Specification Version 1.1, Sun Microsystems, <http://java.sun.com/products/ejb/index.html>]に代表される分散オブジェクト技術やコンポーネント技術（以下、CORBAオブジェクトに代表される分散オブジェクトと、サーバレットやEJBに代表されるコンポーネントをまとめて、「コンポーネント」と記す）や、コンポーネントあるいはそのコンポーネントを内包するプロセスとの間の通信を実行するコンポーネント間通信プラットフォームあるいはプロセス間通信プラットフォーム技術に関わり、特に、分散アプリケーション（以下、「AP」と記す）の試験時や運用時にAPソフトウェアを構成する個々のコンポーネントあるいはサーバプロセス（以下、コンポーネントとサーバプロセスをまとめてサーバと記す）の正常動作性といった信頼性や、処理実行時間、処理実行頻度、処理応答時間といった性能に関するデータの測定・収集とその収集データに基づいたAP全体の信頼性や性能の評価・検証を行い、APの信頼性や性能（以下、信頼性と性能をまとめて品質と記す）のボトルネックを特定し、また、運用時に適切なAP品質を確保するために運用対処を行うAP試験・運用管理技術に関する。

【0002】

【従来の技術】近年、インターネット上での電子商取引

サービスやイントラネットでの社内システムなどの分野を中心に、タイムリーなサービスアプリケーションソフトウェア開発やユーザニーズの変化への迅速な対応の必要性から、コンポーネントソフトウェアを組み上げてアプリケーション（以下 単に「AP」と記す）を構築するAP開発手法を支援し、さらに開発したAPソフトウェアの運用環境への容易な導入を支援するツールへの要望が高まっている。これに対応すべく、WWW(Worldwide Web)、サーバレット、EJB、CORBAなどの技術が注目され、これらの技術を前提とした開発支援ツールやAPサーバ製品の導入が進んでいる。

【0003】Web、サーバレット、EJB、CORBAなどを適用して構築したシステムの典型的な構成を図7に示す。CORBAでの標準の分散オブジェクト間通信プロトコルはIIOP(Internet Inter-ORB Protocol) [Java Servlet API specification, SUN Microsystems Inc., <http://java.sun.com/products/servlet/index.html>]、であり、サーバレットやEJBの仕様を規定するJ2EE(Java2 Enterprise Edition)での標準のコンポーネント間通信プロトコルはRMI (Remote Method Invocation) [Remote Method Invocation, Sun Microsystems Inc., <http://java.sun.com/products/rmi/index.html>]あるいはRMI over IIOPである。IIOP、RMI、RMI over IIOPでのコンポーネント全般の場合のコンポーネント間通信方式を図8に示し、EJBコンポーネント間、及びサーバレット-EJBコンポーネント間の通信の場合コンポーネント間通信方式を図9に示す。

【0004】一般にプロセスやホストを跨るコンポーネント間通信においては、一方のコンポーネントから送出された処理要求は、スタブ、ORB、スケルトンを介してもう一方のコンポーネントに送られ、処理結果は逆にスケルトン、ORB、スタブを介して処理要求発行元のコンポーネントに返される。このスタブ、ORB (Object Request Broker)、スケルトンを介した通信は静的起動通信方式と呼ばれる。CORBA では他に動的起動と呼ばれる通信方式があるが、開発効率や性能の点から、一般に静的通信方式の方が多く用いられる。

【0005】また、EJBコンポーネントに処理要求が送られるときは、スケルトンに加えてEJBコンテナを介する。

【0006】図7にあるように、アプリケーションは、電子商取引サービスなどの一つのサービスを提供するサーバ群 (111-118、141-149)で構成される。分散配備されたAPの個々の構成要素である個々のサーバが他のサーバから処理要求を受け、処理を実行した結果として、サービスユーザに対してサービスが提供される。典型的なAPは、サービスユーザの用いるWebブラウザ(101-105)からHTTPプロトコルを介したサービス要求（以下、HTTPリクエストという）(121-125)を受け必要に応じてユーザ認証処理を行うWebサーバプロセス(111, 115)、

EJBサーバプロセス(112,113,116)、CORBAサーバプロセス(113)、データベース(114,118)、WebサーバプロセスあるいはEJBサーバプロセス内で動作しWebサーバプロセスで受けたHTTPリクエストの内容に従った処理を行うサーバレット(142,146,148)、EJBサーバプロセス中で動作しビジネスロジックや各種サービスに必要な機能を実行するEJBコンポーネント(143,147,149)、及びCORBAサーバプロセス中で動作するCORBAオブジェクト(144)などを含む。WebサーバプロセスがブラウザからHTTPリクエストを受け、WebサーバプロセスあるいはEJBサーバプロセス内で起動されたサーバレットが処理を進め、さらにサーバレットとEJBコンポーネント間あるいはEJBコンポーネント間でRMI、RMI over IIOP、あるいはIIOPプロトコルを介した処理要求(以下、RMI/IIOPリクエスト)を発行しあい(131-136)処理を進めることで、結果として分散配備されたWebサーバプロセス、サーバレット、EJBコンポーネントなど全体で処理を進めてサービスユーザにサービスを提供する。

【0007】また、現在、一般に電子商取引などを提供する事業者側のシステムやデータセンターなどで、サービスの信頼性や応答性を確保するために、現在、システム管理として管理するホストのCPU、メモリ、ハードディスクなどのハードウェアリソースの正常性や使用率を監視することが主に行われている。また、AP管理としてグループウェア、データベースシステム、ERPなど、ソフトウェアとしてひとかたまりで提供され分散化されていないパッケージ製品の動作正常性を監視することが主に行われている。

【0008】

【発明が解決しようとする課題】しかし、Web、サーバレット、EJB、CORBAなどの技術を適用して構築し、分散配備して運用する分散APの信頼性や性能といった品質を測定し、信頼性や処理時間あるいは応答時間が悪化したときに、そのボトルネックとなっているサーバを分散AP構成要素の中から特定することで分散APの品質を評価・検証することを支援し、さらに実装改善や運用対処で品質の保証を図ることを支援するための分散AP試験・運用管理機構は未だ十分確立されていない。

【0009】電子商取引などの競争の激しい分野で他事業者との差別化を図るためには、サービス提供前の試験により品質の評価検証を行い、必要に応じて実装改善や運用対処などにより品質改善を図ることで品質の確保されたサービスをサービスユーザ提供することが有効である。また、サービス導入後でユーザがサービスを利用するときに所定の品質が確保されているかを測定し、必要に応じて実装改善や運用対処を行いサービス品質の確保を図ることが必要である。今後はEJBコンポーネントなど流通しているコンポーネントを組み合わせ、必要に応じて追加拡張して所望のサービスを提供するAPの構築を行うAP開発手法が主流になると予想されるが、コン

ポーネントベースのAPで、サービス利用時にサービスユーザから見たAPの品質が低下したときに品質劣化の主要因となっているコンポーネントやサーバプロセスを特定する機構が未だに確立されていない。現状では、コンポーネント技術を用いてAPを迅速に開発できても、その品質劣化を検出したときに、APソフトウェアに内在するボトルネック個所を特定するのに未だ多くの時間と労力が費やされている。AP試験・運用中におけるサービス品質劣化時にAPを構成するサーバ群の品質の測定を行い、どのコンポーネントやサーバプロセスについて実装改善や運用対処を行えばよいかを判断するための正確な情報を迅速に提供する機構(システム)が望まれる。

【0010】本発明の目的は、APを構成する個々のサーバの品質を測定しデータ収集してAP(構成要素であるサーバ群全体)としての品質の評価・検証を行い、APの品質確保のボトルネックとなるサーバを迅速かつ正確に特定し、必要に応じて自動運用対処することで、APの実行を通して提供されるサービスの品質の改善と確保を支援する機構を実現することにある。

【0011】

【課題を解決するための手段】本発明は 分散オブジェクト技術やコンポーネント技術(以下、分散オブジェクトとコンポーネントをまとめてコンポーネントと記す)に基づき、コンポーネントあるいはそのコンポーネントを内包するプロセス(以下、コンポーネントとプロセスをまとめてサーバと記す)との間の通信を実行するコンポーネント間通信プラットフォームあるいはプロセス間通信プラットフォームを用いる一つ以上のホスト(コンピュータと同義)に分散配備され、一つのサービスを提供する一つ以上のサーバの集合(本明細書中「分散アプリケーション」と定義する)について、サーバに他のコンポーネントあるいはプロセスから処理要求が渡されるとき及びサーバから処理結果が返されるとき、その処理の開始時刻と終了時刻、処理名、及びその処理を実行したサーバ名あるいはそのサーバがコンポーネントである場合はそのクラス名を含む処理性能データを取得して保持し、サーバが他のサーバに処理を要求し処理結果を受けるときに、その処理要求発行時刻と処理結果受信時刻、要求した処理の名前、及びその処理要求発行先のサーバ名あるいはそのサーバがコンポーネントである場合はそのクラス名を含む応答性能データを取得して保持し、その取得した処理性能データと応答性能データから、個々のサーバでの処理実行時間、処理実行頻度、処理要求応答時間などを算出し、それら各々について予め規定された閾値を超過した場合にイベントメッセージを発生させる品質測定部と、任意に選択された一つの分散アプリケーションを構成する個々のサーバに関する処理実行時間、処理実行頻度、処理要求応答時間などのデータを、そのデータを保持する品質測定部群から収集して

提供し、そのデータとアプリケーション構成情報やサーバ間の依存関係情報から分散アプリケーションの性能品質確保のボトルネックとなっている個所を特定し、さらに、任意に選択された一つの分散アプリケーションを構成する個々のサーバで発生した、異常終了や異常状態への遷移などの異常事象に関する、時刻、プロセスID、コンポーネント名、コンポーネントクラス名などの情報を収集して提供し、分散アプリケーションの信頼品質確保のボトルネックとなっている個所を特定する品質データ収集分析部とを用いて、APを構成する個々のサーバの品質測定とアプリケーションとしての品質評価検証を行い、APの品質確保のボトルネックとなっているサーバを迅速かつ正確に特定する。

【0012】また、本発明のさらに別の態様によれば、必要に応じて自動運用対処するように試験・運用管理実行条件管理及び試験・運用管理コーディネータ部を備えていることを特徴とする。

【0013】

【発明の実施の形態】以下、本発明の実施の形態を説明する。

【0014】図1は本発明にかかる分散AP試験・運用管理装置の各構成要素間の関係を表す図であり、図2はその運用時のプロセス構成を示す図である。本発明の実施の形態にかかる分散AP試験・運用管理装置は、以下の構成部を有する。なお 図1、2において説明の便宜上同一構成要素に異なる参照符号を附したが これは構成要素が異なることを意味するものではない。

【0015】(1) GUI部(図1、201、図2、301)
AP試験実行者や運用管理者からAP操作要求を受け、試験結果データや運用管理状況データを表示する。

【0016】(2) APライフサイクル管理部 (図1、202、図2、305、307、308、311、314、317)
以下の機能を有する。

【0017】

(2-1) APの構成情報と配備情報(図1、231)の保持
APのコンポーネント構成情報を保持する。また、それら個々のコンポーネントについて、どのホストで動作しているか(及び動作可能であるか)についての配備情報を保持する。

【0018】(2-2)サーバエントリとAPエントリ情報(図1、232)の保持
サーバプロセスの起動や停止を行うために必要な情報をサーバエントリとして保持し、さらに、サーバの集合であるAPの起動や停止を行うために必要な情報を保持するAPエントリとして保持する。サーバエントリには、サーバプロセスの動作ホスト名、実行ファイルの絶対パス名、サーバプロセス起動コマンドなどが含まれる。APエントリは、サーバエントリの集合として設けられ、APの構成要素であるサーバ群の起動や停止を行うために用いられる。

【0019】(2-2)サーバプロセスの(再)起動と停止

(2-3)サーバプロセスの動作状態監視

サーバプロセスの動作状態を監視し、起動、正常終了、異常終了、異常状態などの状態変化を検出したときにイベントメッセージを発生・送出する。プロセスの起動は、後述の品質測定部の起動時に送出されるメッセージをAPライフサイクル管理部が受けることで検出する。プロセスの正常終了は、オペレーティングシステムが提供するシグナルやシステムコールなどを用いて検出する。プロセスの異常終了は、アプリケーションライフサイクル管理部からサーバプロセスへのポーリングの異常や、オペレーティングシステムが提供するシグナルやシステムコールなどを用いて検出する。プロセスの異常状態は、サーバプロセスへのポーリングの異常から検出する。また、単位時間あたりのアプリケーション構成要素のいずれかのサーバプロセスでの異常終了や異常状態の発生の回数をアプリケーションの異常発生頻度として定期的に計測し、異常発生頻度の少なさを動作正常性として扱う。

【0020】(2-4)APの(再)起動と停止

(2-5)試験用クライアント(図1、213、図2、310)の起動と停止

アプリケーションの品質試験を行うときに、試験・運用管理コーディネータ部(図1、202、図2、304)から試験操作の指示を受けて、試験・運用管理実行条件管理部(図1、203、図2、303)の保持する試験実行条件情報(図1、237)に従って試験用クライアント部(図1、213、図2、310)を起動し、サービスユーザの用いるブラウザ(図2、212、図331)の代わりにWebサーバプロセス(図1、241、244、図2、341)に対して試験用のリクエストを発生させる。また、試験終了後に起動した試験用クライアント部を停止終了させる。

【0021】(3) 試験・運用管理実行条件管理部(図1、203、図2、303)

以下のデータを格納し要求者に提供する。

【0022】

(3-1) コンポーネント管理オプション(図1、233)

どのコンポーネントに対してポーリングを行うか、どのコンポーネントの処理実行時間や処理実行頻度などの性能に関するデータについてどの程度の時間間隔で収集するか、各々のコンポーネントの処理実行時間や処理実行頻度といった性能について許容される限界値(閾値)はどの程度か、どのコンポーネントについて処理実行時間や処理実行頻度などが予め規定された閾値を超えた場合にイベントメッセージを発生・送出させるかなどについてのデータ。

【0023】(3-2) ホスト管理オプション(図1、234)
どのホストについてCPU、メモリ、ハードディスクといったハードウェア資源の使用率のデータ収集を行うか、各々のホストについてそれら各々のリソース使用率の許

容される限界値(閾値)はどの程度がなどについてのデータ。

【0024】(3-3) 試験・運用管理ポリシ(図1、235) A P試験・運用管理時に生成され送出されたイベントメッセージを受け、そのイベントメッセージの内容に応じて自動的に望ましいA P運用対処を本発明にかかる装置が自動で実行するために用いられるデータ。検出イベント種別、イベント内容に関する対象ホストと対象サーバ、及び対処手続きから構成されるポリシエレメントの集合として表される。

【0025】検出イベント種別の例として以下が挙げられる。

【0026】コンポーネントの起動、コンポーネントの正常終了/異常終了、コンポーネントのポーリング異常、コンポーネント性能閾値オプションの各項目における閾値超過の、指定された回数の超過、CPU使用率などのハードウェアリソースの、指定された値の超過、ポリシに従った自動対処手続きの実行の失敗。

【0027】また、対処手続きは対処手続き要素を組み合わせで定義される。対処手続き要素の例としては以下が挙げられる。A P(再)起動、A P停止、A P閉塞/閉塞解除(一時停止/再開)、依存関係にあるコンポーネントの停止、サーバプロセス(再)起動、サーバプロセス停止、サーバ実行ファイルの指定ホストの指定ディレクトリへの配備、コンポーネント閉塞/閉塞解除(一時停止/再開)、本発明にかかる装置の構成要素へのイベント内容の通知、本発明にかかる装置の構成要素への実行した対処手続き内容及び対処手続き実行成功/失敗の通知、CPU使用率の最も低いホストの選択。

【0028】(3-4) レポートフォーマット(図1、236) テキスト形式及びHTML形式に加工され格納される、処理実行時間、処理実行頻度、各種閾値、処理性能の保証率(処理実行時間が閾値以下の処理数/全処理数)、及び応答性能の保証率(応答時間が閾値以下の処理要求数/全処理要求数)などの品質測定結果データの表示フォーマットを指定するデータ。

【0029】(3-5) 試験実行条件(図1、237) 試験実行時のWebサーバプロセスに対してHTTPリクエストを発行する試験用クライアントを動作させるホスト群、それらの各ホストで動作させる試験用クライアントの数、及び各々の試験用クライアントでのリクエスト発生時間間隔など、試験を実行するにあたっての試験環境設定条件を規定するデータ。

【0030】

(3-6) コンポーネント依存関係(図1、238) コンポーネント間の処理呼び出し関係に基づき定義され、「依存元コンポーネント名-依存先コンポーネント名」の対の集合で表される。特定のA Pに依存する形でも、A Pに依存しない形でも定義できる。特定のA Pに依存する形で定義した場合には、そのA Pが起動されて

いる場合にのみその依存関係が適用される。2つのコンポーネントが依存関係にある場合、依存先のコンポーネントの停止が検出された場合に依存元のコンポーネントも強制停止させられる。

【0031】(4) 試験・運用管理コーディネータ部(図1、202、図2、304)

A P試験実行者から試験実行の指示を受け、A Pライフサイクル管理部(図1、202、図2、308)に対して試験用クライアント部の起動を指示したり、A Pを構成する各々のコンポーネントから信頼性や性能のデータを収集することを品質データ収集分析部(図1、202、図2、306)に対して指示したりして、試験手続きを進める

また、A Pライフサイクル管理部(図1、202、図2、308)や後述の品質測定部(図1、205-210、図2、313,316,319)など本発明にかかるシステムの他の構成要素から異常発生等のイベントメッセージを受けたときに、上記の試験・運用管理実行条件管理部(図1、203、図2、303)の保持する試験・運用管理ポリシ情報(図1、235)を参照し、該当するポリシエレメントの内容に従った対処手続きを実行する。

【0032】

(5) 試験入力データ生成部(図1、204、図2、302) Webサーバプロセス(図1、241,244、図2、341)が扱うHTMLファイル群の内容からHTTPリクエスト発行に關与するURL(Unified Resource Locator)を検索・抽出し、さらに試験実行者から与えられた入力引数データを参照して試験用URLのリスト(図1、251)を生成する。

【0033】

(6) 試験用クライアント部(図1、213、図2、310) 試験入力データ生成部(図1、204、図2、302)が生成した試験用URLリスト(図1、251)を参照してWebサーバプロセス(図1、241,244、図2、341)に対してHTTPリクエストを発行する。

【0034】

(7) 品質測定部(図1、205-210、図2、313,316,319) 試験用クライアント部(図1、213、図2、310)から試験用HTTPリクエストが発行され、それに続きA P実行にあたりWebサーバプロセス、サプレット、EJBコンポーネントなどのサーバへ処理要求が渡されたとき及び処理結果が返されるとき(測定タイミングは、後述のプリプロセッサ部が品質測定用のソースコードを埋め込んだ場所に依存する)、それらのサーバ実行される個々の処理についての処理開始時刻、処理終了時刻、処理実行サーバ名、処理名(メソッド名)などの処理性能データを取得し保持する。このとき、処理終了時刻から処理開始時刻を引くことにより処理実行時間を算出する。

【0035】さらに、単位時間あたりの実行開始した処理の数を処理実行頻度として定期的に算出する。処理実行時間や処理実行頻度が予め定められた閾値を超えた場合、イベントメッセージを生成し送出する。

【0036】また、アプリケーション実行にあたりWebサーバプロセス、サーブレット、EJBコンポーネントなどの他のサーバへ処理要求が発行するとき及び処理結果を受けるときに(測定タイミングは後述のプリプロセッサ部が品質測定用のソースコードを埋め込んだ場所に依存する)、要求する処理についての処理要求発行時刻、処理結果受信時刻、要求処理名、処理要求先サーバ名などの応答性能データを取得・保持し、処理結果受信時刻から処理要求発行時刻を引いた値を処理応答時間として算出する。処理応答時間が予め定められた閾値を超えた場合、イベントメッセージを生成し、送出する。

【0037】また、品質測定部(図1、205-210、図2、313,316,319)は、自身の起動時にAPライフサイクル管理部(図1、202、図2、305,307,308,311,314,317)と品質データ収集分析部(図1、202、図2、306)に対して、自身へ処理要求を送るために必要なリファレンス情報を送出する。これ以降、APライフサイクル管理部から品質測定部に状態確認のためのポーリングを行った後、品質データ収集分析部がその品質測定部から性能データを収集したりすることが可能となる。

【0038】

(8) 品質データ収集分析部(図1、202、図2、306)
品質測定部が取得したデータから、個々のサーバでの処理実行時間(その履歴と平均)、処理実行頻度などのデータを作成する。さらに、AP構成情報とコンポーネント依存関係情報を用いてAP構成要素サーバ間の処理呼び出し関係を調べ、その関係情報を用いてコンポーネントの集合、すなわちAP全体としての信頼性や性能といった品質のボトルネック箇所を特定する。

【0039】

(9) レポート作成部(図1、202、図2、306)
品質データ収集部が収集したAPの信頼性や性能のデータを基にテキスト形式やHTML形式でレポートを作成する。

【0040】(10) ホスト運用データ取得部(図1、202、図2、309,312,315,318)

CPU、メモリ、ハードディスクといったホストのリソースの使用率を定期的に調べる。それらのリソース使用率のいずれかが予め定められた閾値を超えた場合、イベントメッセージを生成し送出する。

【0041】

(11) ホスト運用データ収集部(図1、202、図2、307)
定期的に、あるいは本発明の他構成要素からの要求に応じて、各ホストにおけるCPU、メモリ、ハードディスクといったリソースの使用率のデータをホスト運用データ取得部(図1、202、図2、309,312,315,318)から収集して提供する。

【0042】(12) プリプロセッサ部(図1、211)

上記の品質測定部(図1、205-210、図2、313,316,319)をアプリケーション開発者の作成した各種サーバ実行フ

ァイルに組み込む。

【0043】以上の構成部を用いて、上記目的を実現する。

【0044】[実施の形態の動作] AP試験・運用管理者は、以下の実施手順に従って分散APの試験・運用管理を行う。

【0045】

(1) サーバ実行ファイルへの品質測定部の組み込み
AP開発者あるいはAP試験・運用管理者は、AP試験・運用管理を行うにあたり、まず、AP開発者が作成したソースコードファイル群や分散APプラットフォームあるいはAPサーバが自動生成したソースコードファイル群へ品質測定部を埋め込み、その後にそれらのソースコードファイルをコンパイラに通して、サーバプロセスを起動・動作させるために必要なサーバ実行ファイルを作成する。一例として、Java言語で作成したサーブレットやEJBコンポーネントのソースコードに品質測定部を埋め込み、コンポーネントの品質測定を可能とするためのサーバ実行ファイルを作成する手順を図3に示す。また、サーバにおける処理開始時刻あるいは処理要求発行時刻を取得するために埋め込まれるソースコードの一例を図4に、処理終了時刻あるいは処理結果受信時刻を取得するためのソースコードの一例を図5に示す。また、2つのソースコード埋め込み方法の例を図6に示す。

【0046】この手順に従いサーブレットの処理性能を計測するための品質測定部を埋め込む場合は、(a) AP開発者の作成したサーブレットのソースコードでの各々のメソッド記述の先頭部分に処理開始時刻を取得するためのソースコードを挿入するとともに、最終部分に処理終了時刻を取得するためのソースコードを挿入する(図6中の「パターン1」の方法)か、あるいは(b)スケルトンのソースコードがAP開発者の作成したコードを呼び出す直前の部分に処理開始時刻を取得するためのソースコードを挿入するとともに、直後の部分に処理終了時刻を取得するためのソースコードを挿入する(図6中の「パターン2」の方法)。その後にコンパイラを用いてサーバ実行ファイルを作成する。

【0047】また、この手順に従いEJBコンポーネントの処理性能を計測するための品質測定部を埋め込む場合は、(c) アプリケーションサーバ製品付属のツールが作成したソースコードファイルのEJBコンテナ記述中にある、EJBコンポーネントのメソッドを呼び出す記述の直前に処理開始時刻を取得するためのソースコードを、直後に処理終了時刻を取得するためのソースコードを挿入する(図6の「パターン2」の方法)か、(d) スケルトンのソースコードがEJBコンテナあるいはEJBコンポーネントを呼び出す直前の部分に処理開始時刻を取得するためのソースコードを、直後の部分に処理終了時刻を取得するためのソースコードを挿入する(図6中の「パターン2」の方法)か、あるいは(e) AP開発者の作成したEJBコン

ポーネントのソースコードでの各々のメソッド記述の先頭部分に処理開始時刻を取得するためのソースコードを、最終部分に処理終了時刻を取得するためのソースコードを挿入する(図6中の「パターン1」の方法)。その後コンパイラを用いてサーバ実行ファイルを作成する。

【0048】また、この手順に従いサプレットが要求した処理に関する応答性能を計測するための品質測定部を埋め込む場合は、(a) AP開発者の作成したサプレットのソースコードで処理要求を起動する直前に処理要求発行時刻を取得するためのソースコードを埋め込むとともに、直後に処理結果受信時刻を取得するためのソースコードを埋め込む(図6中の「パターン2」の方法)か、あるいは(b)EJBコンポーネントが呼び出すスタブのソースコードでの各メソッド記述の先頭部分に処理開始時刻を取得するためのソースコードを、最終部分に処理終了時刻を取得するためのソースコードを挿入する(図6中の「パターン1」の方法)。その後コンパイラを用いてサーバ実行ファイルを作成する。

【0049】この手順に従いEJBコンポーネントが要求した処理に関する応答性能を計測するための品質測定部を埋め込む場合は、(a) AP開発者の作成したEJBコンポーネントのソースコードで処理要求を起動する直前に処理要求発行時刻を取得するためのソースコードを、直後に処理結果受信時刻を取得するためのソースコードを埋め込む(図6中の「パターン2」の方法)か、あるいは(b)EJBコンポーネントが呼び出すスタブのソースコードでの各メソッド記述の先頭部分に処理開始時刻を取得するためのソースコードを、最終部分に処理終了時刻を取得するためのソースコードを挿入する(図6中の「パターン1」の方法)。その後コンパイラを用いてサーバ実行ファイルを作成する。

【0050】

(2) サーバエントリ及びAPエントリの設定

AP試験・運用開発者は、GUI部を介してサーバエントリとAPエントリをAPライフサイクル管理部に登録する(図2、361)。

【0051】(3) コンポーネント監視条件や性能データ取得条件の設定

AP試験・運用開発者は、GUI部を介してコンポーネント管理オプション情報を試験・運用管理実行条件管理部に登録する(図2、361)。

【0052】(4) 試験・運用管理ポリシーの設定

AP試験・運用開発者は、GUI部を介して試験・運用管理ポリシー情報を試験・運用管理実行条件管理部に登録する(図2、361)。

【0053】(5) AP起動

AP試験・運用開発者は、GUI部でAPエントリ名を指定し、試験・運用管理コーディネータ部を通してドメインマネージャに存在するAPライフサイクル管理部にA

P起動を要求する(図2、361)。ドメインマネージャに存在するAPライフサイクル管理部は、指定された名前のAPエントリを参照し、そのAPエントリを構成する個々のサーバエントリを確認し、それらのサーバエントリ毎に、サーバマネージャに存在するAPライフサイクル管理部にサーバプロセスの起動を要求する(図2、363、364、366、368)。サーバマネージャに存在するAPライフサイクル管理部は、指定されたホストで指定されたサーバ実行ファイルを用いてサーバプロセスを起動する(図2、377-379)。サーバプロセスが起動されたとき、この手続きによりAPを構成するサーバがすべて起動され、結果的にAPが起動されたことになる。

【0054】(6) 試験用クライアント起動

APの試験を行うにあたり、試験実行者は、GUI部を介して試験・運用管理コーディネータ部に試験用クライアントの起動を要求する(図2、361)。試験・運用管理コーディネータ部は、試験・運用管理実行条件管理部から試験実行条件を入手し、その内容を基に、指定されたホストで指定された数の試験用クライアントを起動するよう、サーバマネージャに存在するAPライフサイクル管理部に要求する(図2、363)。起動された試験用クライアントは、試験用HTTPリクエストを指定された時間間隔でWebサーバプロセスへ送出する(図2、380)。

【0055】(7) 性能データ取得

Webサーバプロセスが試験用クライアント(試験時)やサービスユーザが用いるブラウザ(運用時)からHTTPリクエストを受ける(図2、370、380)と、WebサーバプロセスあるいはEJBサーバプロセスで起動されたサプレットからEJBコンポーネントへ(図2、371)、あるいはEJBコンポーネントから他EJBコンポーネントへRMI/IIOPリクエストが送られ(図2、372)、その結果コンポーネント全体でサービス実行処理が進められる。各コンポーネントにて要求された処理が実行されるとき、品質測定部にて、各コンポーネントで実行した処理について処理開始時刻、処理終了時刻、処理名(メソッド名)、処理実行サーバ名あるいはそのサーバがコンポーネントである場合はそのクラス名を含む処理性能データが取得され(図2、373-375)、そのデータを基に処理実行時間(履歴、平均とも)や処理実行頻度などのデータが算出され蓄積される。また、各コンポーネントが他のコンポーネントに処理の実行を要求し処理結果を受けるときに、品質測定部にて、各コンポーネントから発行した処理要求について、その処理要求発行時刻と処理結果受信時刻、処理名(メソッド名)、及びその処理要求発行先のサーバ名あるいはそのサーバがコンポーネントである場合はそのクラス名を含む応答性能データが取得され(図2、373-375)、そのデータを元に応答時間(履歴、平均とも)などのデータが算出され蓄積される。

【0056】(8) 異常発生時の自動対処

AP試験時や運用時に、プロセス異常停止やコンポーネ

ントでの処理実行時間や処理実行頻度の閾値超過などの異常がサーバマネージャ内のライフサイクル管理部(図2、309、312、315、318)によって検出されたとき、その旨のイベントメッセージが試験・運用管理コーディネータ部に送出される(プロセス異常停止の場合は図2、365,367,369、閾値超過の場合は図2、389-391)。試験・運用管理コーディネータ部は、試験・運用管理実行条件管理部で保持されている試験・運用管理ポリシー情報を参照して、対処手続きを実行する。

【0057】一例として、プロセス異常停止の自動対処の実行例は次のようになる。まず、サーバマネージャに存在するAPライフサイクル管理部がWebサーバプロセスやEJBサーバプロセスなどでのプロセス異常停止を検出する(図2、377-379)。次に、サーバマネージャに存在するAPライフサイクル管理部は、ドメインマネージャに存在するAPライフサイクル管理部を通して試験・運用管理コーディネータ部にその旨のイベントメッセージを送出する(図2、365,367,369)。試験・運用管理実行条件管理部の保持する試験・運用管理ポリシー情報から発生したイベント(この場合は「サーバ異常終了」)に関係するポリシーエレメントデータを検索する。ここで、該当するポリシーエレメントデータが見つからなかった場合は何も対処手続きを実行しない。該当するポリシーエレメントデータが見つかった場合には、そのポリシーエレメントに記載されている内容に従って対処手続きを進行させる。

【0058】例えば、対処手続きの内容が下記(a)ー(c)であった場合、下記の手順で手続きが実行される。

【0059】(a)CPU使用率の最も低いホストの選択
ホスト運用データ収集部から運用中の各ホストのCPU使用率情報を収集し、最も使用率の低いホストを選択する。

【0060】

(b)選択されたホストへのサーバ実行ファイルの配備
AP試験・運用管理者の指定したホストから、上記(a)の手続きにより選択されたホストの所定のディレクトリへサーバ実行ファイルを配備する。

【0061】

(c)選択されたホストでのサーバプロセスの起動
選択されたホストの所定のディレクトリにあるサーバ実行ファイルを用いてサーバプロセスを起動する。

【0062】

(9) 信頼性・性能データ集計とレポート作成
AP試験・運用管理ポリシーを適用してAPの試験・運用を行っているときに、品質データ収集部が、品質データ測定部からAPを構成する個々のコンポーネントの性能データを収集し(図2、385-388)、APライフサイクル管理部からAPを構成するサーバプロセスの異常発生頻度データを収集する(図2、392)。さらに、レポート作成部が、品質データ文書部が収集したデータを基に通信

信頼性や性能のレポートを作成する。

【0063】

【発明の効果】以上説明したように、本発明によれば、サーバの集合体であるAP単位での試験・運用管理を行うことが可能となる。また、分散APの構成要素であるサーバプロセスやコンポーネント単位で信頼性や性能を測定し評価することが可能になり、さらにアプリケーションの信頼性や性能を確保する上でのボトルネックとなるサーバプロセスやコンポーネントを特定することが可能となる。

【0064】また、試験入力データ生成部や試験用クライアント部などを用いることで、試験の自動実行が可能となり、手動で行う場合に比べ試験実行の負担が軽減される。

【図面の簡単な説明】

【図1】本発明の構成要素と構成要素間の関係を説明する図である。

【図2】本発明のどの構成要素がどのプロセス上で動作するかを説明する図である。

【図3】サーバ実行ファイルへ本発明の一構成要素である品質測定部を組み込む手順の一例を説明する図である。

【図4】品質測定部の一部として処理開始時刻取得用及び処理要求発行時刻取得用に埋め込まれるソースコードの一例を示す図である。

【図5】品質測定部の一部として処理終了時刻取得用及び処理結果受信時刻取得用に埋め込まれるソースコードの一例を示す図である。

【図6】品質測定部のソースコードをサーバ実行ファイルへ埋め込む方法例を示す図である。

【図7】Web、サーブレット、EJB、CORBAなどを適用して構築したシステムの典型的な構成を示す図である。

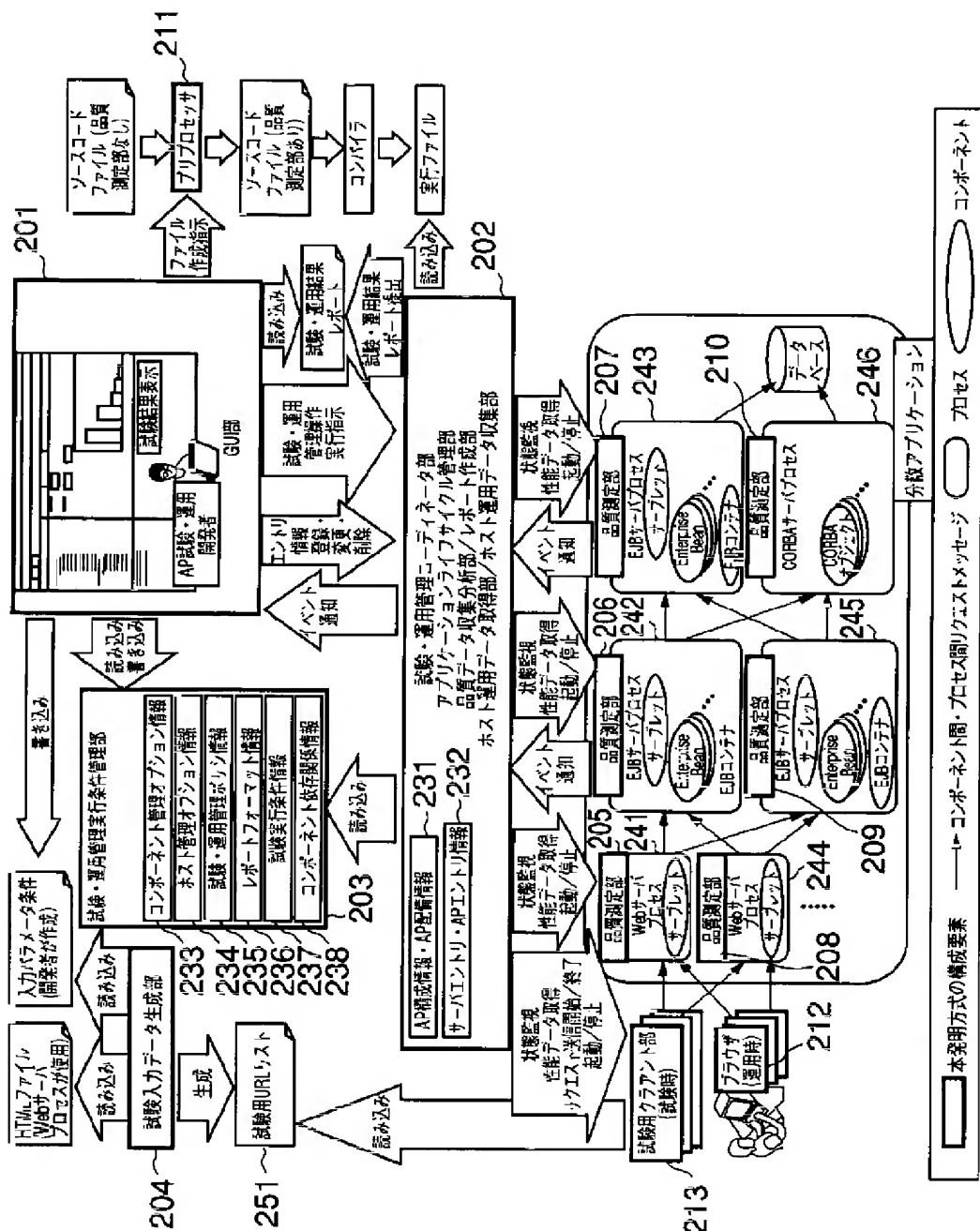
【図8】COBRAオブジェクト、EJBコンポーネント、サーブレットなどのコンポーネントが相互に通信する方式で、静的起動の場合を説明する図である。

【図9】COBRAオブジェクト、EJBコンポーネント、サーブレットなどのコンポーネントが相互に通信する方式で、静的起動の場合を説明する図である。

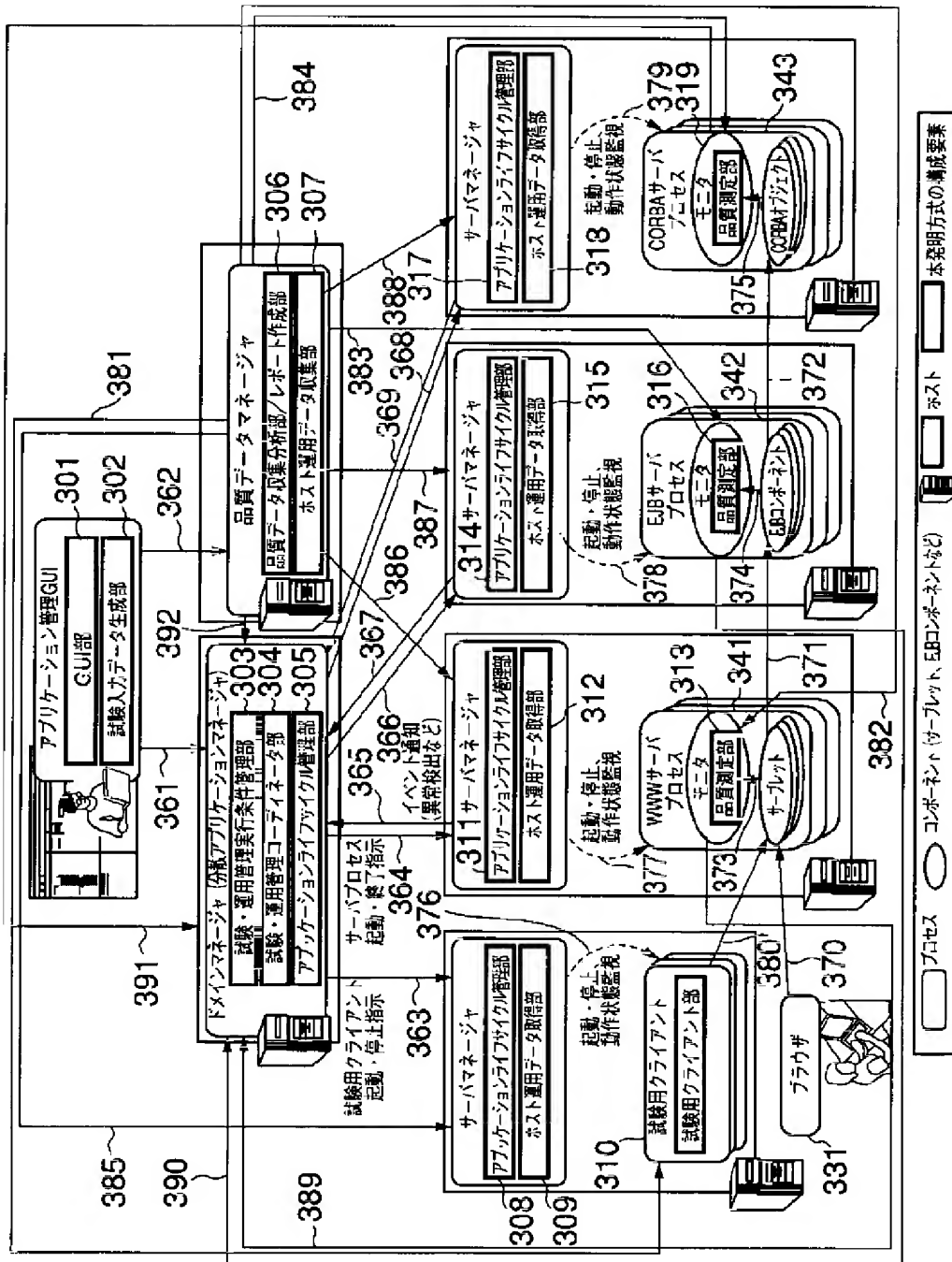
【符号の説明】

101~105 … ブラウザ
111,115 … Webサーバプロセス
112,116,117 … EJBサーバプロセス
113 … CORBAサーバプロセス
114,118 … データベースサーバ
121~125 … HTTPリクエスト
131,132,134,135 … IIOP,RMIあるいはRMI over IIOPを用いた処理要求
133,136 … データベースアクセス用通信プロトコルを用いた処理要求
141,142,145,146,148 … サーブレット

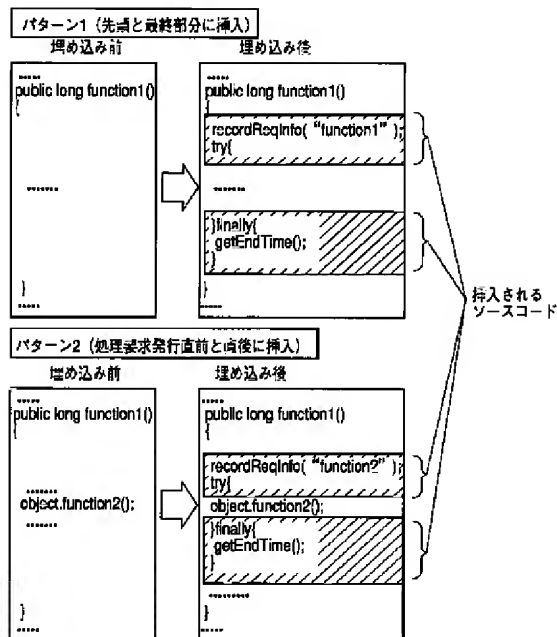
【図1】



【図 2】



【図6】



【図7】

